Exploiting Usage to Predict Instantaneous App Popularity : Trend Filters and Retention Rates

Sigg, Stephan

2019-04

Sigg , S , Lagerspetz , E , Peltonen , E , Nurmi , P & Tarkoma , S 2019 , ' Exploiting Usage to Predict Instantaneous App Popularity : Trend Filters and Retention Rates ' , ACM Transactions on the Web , vol. 13 , no. 2 , 13 . https://doi.org/10.1145/3199677

http://hdl.handle.net/10138/303128
https://doi.org/10.1145/3199677

acceptedVersion

Exploiting usage to predict instantaneous app popularity: Trend filters and retention rates

STEPHAN SIGG, Aalto University, Finland
EEMIL LAGERSPETZ, University of Helsinki, Finland
ELLA PELTONEN, Insight Centre for Data Analytics, University College Cork, Ireland
PETTERI NURMI, University of Helsinki, Finland and Lancaster University, United Kingdom
SASU TARKOMA, University of Helsinki, Finland

Popularity of mobile apps is traditionally measured by metrics such as the number of downloads, installations, or user ratings. A problem with these measures is that they reflect usage only indirectly. Indeed, retention rates, i.e., the number of days users continue to interact with an installed app, have been suggested to predict successful app life-cycles. We conduct the first independent and large-scale study of retention rates and usage trends on a dataset of app-usage data from a community of 339,842 users and more than 213, 667 apps. Our analysis shows that, on average, applications lose 65% of their users in the first week, while very popular applications (top 100) lose only 35%. It also reveals, however, that many applications have more complex usage behaviour patterns due to seasonality, marketing, or other factors. To capture such effects, we develop a novel app-usage trend measure which provides instantaneous information about the popularity of an application. Analysis of our data using this trend filter shows that roughly 40% of all apps never gain more than a handful of users (*Marginal* apps). Less than 0.1% of the remaining 60% are constantly popular (*Dominant* apps), 1% have a quick drain of usage after an initial steep rise (*Expired* apps), and 6% continuously rise in popularity (*Hot* apps). From these, we can distinguish, for instance, trendsetters from copycat apps. We conclude by demonstrating that usage behaviour trend information can be used to develop better mobile app recommendations.

CCS Concepts: • **Information systems** **Information retrieval**; *Content analysis and feature selection*; *Users and interactive retrieval*; *Probabilistic retrieval models*; *Specialized information retrieval*;

Additional Key Words and Phrases: Mobile Analytics; Application Popularity; Trend Mining

ACM Reference Format:
Stephan Sigg, Eemil Lagerspetz, Ella Peltonen, Petteri Nurmi, and Sasu Tarkoma. 2010. Exploiting usage to predict instantaneous app popularity: Trend filters and retention rates. *ACM Trans. Web* 9, 4, Article 39 (March 2010), 24 pages. https://doi.org/0000001.0000001

1 INTRODUCTION

With the popularity of mobile apps continuing rapid growth, judging on the potential of an individual app has become far from straightforward. Studies on app marketplaces have shown that overall rating and the nature of user reviews are key drivers in application download decisions [20, 25]. Recommendations, though, are biased towards apps with a large user base. High-potential successful

Authors' addresses: Stephan Sigg, Aalto University, Department of Communications and Networking, Otakaari 5 a, Espoo, Finland; Eemil Lagerspetz, University of Helsinki, Department of Computer Science, Helsinki, Finland; Ella Peltonen, Insight Centre for Data Analytics, University College Cork, Cork, Ireland; Petteri Nurmi, University of Helsinki, Department of Computer Science, Helsinki, Finland , Lancaster University, Lancaster, United Kingdom; Sasu Tarkoma, University of Helsinki, Department of Computer science, Helsinki, Finland.

1559-1131/2010/3-ART39 $15.00
https://doi.org/0000001.0000001

apps with a smaller community of users, have a higher risk to vanish in this digital cornucopia. Indeed, rating systems in current app stores have been shown to foster an effect of accumulated advantage (so-called Matthew effect) whereby popular apps increase in popularity while less popular ones dwindle [45]. Furthermore, ratings are vulnerable to spam and rating fraud [11, 22] and downloads can be misleading since users install apps simply to try them out [3, 6] and rate apps negatively for a wide range of reasons, such as technical problems or lack of features [24].

Retention rate has been proposed as a usage-based metric to measure the success of mobile apps[1]. Retention rate reflects the number of days that users continue to interact with an application after installing it. Reports[2] suggest that retention rates of most applications are low, i.e. actual usage of apps differs considerably from installations and download counts. However, thus far no independent information about retention rates of mobile apps has been published.

As our **first contribution**, we present results from the first independent study of retention rates in the wild. Previous reports on retention rates have solely been published by analytics companies, which may bias the reported values towards the (commercial) interests of these companies. Verifying the accuracy and generality of these reports through an independent source is thus essential for characterising and understanding retention patterns of mobile apps[1,2]. We perform our analysis on usage data recorded in the frame of the Carat project [33] from 339,842 Android devices over a period of three years (June 2012 – July 2015). Our results indeed confirm that, on average, applications lose 70% of their users within a week from first use. However, these effects are mediated by the number of users an application has. For applications with 10 or more users, first day retention rates below 30% are rare, and retention rates increase with better known apps. However, we also show that retention rates alone are not a sufficient measure of an application's success as they ignore fluctuation in instantaneous usage, effects of seasonality, external factors, and other long-term usage behaviour trends.

As our **second contribution**, we present a novel app-usage trend filter which can capture and quantify these effects. It describes the relative popularity of apps based on daily use, indicates behaviour trends regardless of absolute volume and categorises apps into *App trend patterns* that predict the relevance of an app. This type of fine-grained trend information can provide developers feedback about instantaneous popularity of their application, so that they can apply timely changes as countermeasure. Trend information can also be used for marketplace analytics to support advertising, or, on the other hand, trend status can be used as an additional metric to clean the store of irrelevant copycat apps.

We validate our app-usage filter through a large-scale analysis of mobile app usage trends. Our results show that 40% of all apps never ever acquire more than a handful of users (*Marginal* apps), and in the remaining 60%, only 0.1% are popular over a continuous period (*Dominant*), 1% are drastically drained in their usage after an initial steep rise (*Expired*), and 6% are continuously rising in popularity (*Hot*). The remaining applications showed only weak correlation with the most distinctive trend patterns and hence were not associated with any specific pattern.

As a **practical use case**, we further analyse the performance of a state-of-the-art mobile app recommender [48] with respect to trends. Our analysis shows that only 3.6% of the recommendations are for apps which are currently rising in popularity, and that overall recommendations have low novelty and temporal diversity. We also demonstrate that the accuracy of the recommendations can be improved by considering trend information.

[1] http://info.localytics.com/blog/the-8-mobile-app-metrics-that-matter
[2] http://andrewchen.co/new-data-shows-why-losing-80-of-your-mobile-users-\is-normal-and-that-the-best-apps-do-much-better/

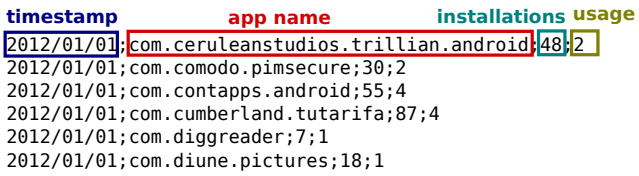


Fig. 1. App-usage data from Android: process name, installation count, and daily usage.

2 DATASET

Our work considers data recorded with Carat [33][3], a stock Android and iOS mobile app designed to offer personalised recommendations to improve a device's battery life. Carat uses energy-efficient and non-invasive instrumentation to record the state of the device, including the process list, and active apps. Carat has been deployed on over 800,000 smartphones, roughly half of which are Android devices. The community of users that contribute data to Carat is spread all over the world, with users in roughly 200 countries, and a strong presence in USA, most of Europe, India, and Japan.

For this paper, we consider a subset of the data covering a period of three years (June 2nd, 2012 – July 14th, 2015) that contains measurements from 339,842 Android devices[4]. We limit our analysis to Android devices as the data obtained on Android devices can be uniquely mapped into individual applications [42], whereas data obtained on iOS devices requires more complicated processing since the app names obtained from iOS in Carat are presented by IDs only, which differ for distinct versions of the same app. As part of our analysis, we assess the popularity of apps within categories. To carry out this analysis, we combine the crowdsourced data with category information from Google Play. The resulting dataset includes a total of 13,779,666 app usage records from 213,667 apps in 47 categories. We only use the leaf categories of Google Play, such as *Games: Racing* and *Family: Action*. The scale of the dataset we consider in our analysis is an order of magnitude larger than in previous works. For example, Harman et al. [20] considered a dataset containing reviews from 30,000 apps, whereas Böhmer et al. [5] considered measurements from 4,125 users and 22,626 applications. Our proposed approach estimates the popularity of apps with respect to a given period (June 2nd, 2012 – July 14th, 2015). Results achieved for other measurement periods might differ as a result of different life-cycle states of an app.

Figure 1 details a small sample of the data we utilised for our study on usage trends. Note that only anonymized data is exploited (e.g. timestamp, app name, usage) and that the installation count was necessary only for our comparison to a state-of-the-art recommendation system in Section 6. For the calculation of retention rates in Section 3.1, anonymized user IDs have been exploited in addition. Consequently, in contrast to other methods exploiting personal information and usage information, our approach is applicable at large scale. Usage frequency data is potentially discontinuous (especially for apps resembling the *Marginal* pattern, introduced in Section 4), and absolute values of different apps vary. To meaningfully compare app usage trends, normalisation with respect to the total usage count (relative popularity over time) and within maximum usage of the app (popularity within a particular day) is therefore necessary. For apps with low user base, intermediate zero-usage days occur (discontinuous usage patterns). These events are interpolated during data pre-processing.

We identified the following potential limitations due to the particular data set utilised. First, data was collected using a custom mobile application, which itself is prone to the studied usage trends and retention. As the Carat application has been designed to support energy-awareness, there is an inherent bias towards users interested in their smartphones' energy consumption. Carat is only

[3] carat.cs.helsinki.fi

[4] the dataset is available via [5]. For further support, please contact carat@cs.helsinki.fi

available in three languages (Finnish, English, Italian), and hence the sample is likely biased to people with sufficient knowledge of one of these languages. Furthermore, Carat collects measurements continuously in the background of the mobile device, but only sends the data when launched. Therefore, we obtain ample data to carry out our analysis as long as the user launches the application *once* after a sufficiently long period from initial use. To minimise this effect, for our results on general properties of the dataset, we considered all uploads in the period (June 2nd, 2012 – July 14th, 2015) made until June 2nd 2017. To further limit potential biases caused by users stopping to use Carat, we only considered users who had used Carat over a sufficiently long period, i.e., a month.

While comparing the popularity of apps within a category, we relied on category information extracted directly from Google Play. On Google Play, the categorisation of an app is the responsibility of the developer, and consequently similar apps are likely to contain variations in their categorisations. An alternative would be to rely on topic models to derive a categorisation of the apps. For instance, Gorla et al. [17] have demonstrated the use of Latent Dirichlet Allocation (LDA) for mining categories from app store data.

Ethical Considerations: We analyse aggregate-level data which contains no personally identifiable information. The privacy protection mechanisms of Carat are detailed in [33]. Data collection by Carat is subject to the IRB process of University of California, Berkeley. Users of Carat are informed about the collected data and give their consent to use data from their devices.

3 ANALYSIS OF RETENTION RATES

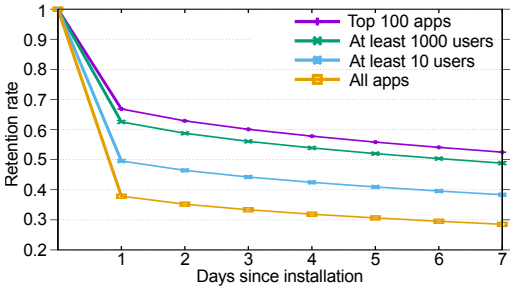
Existing academic studies on mobile app usage have characterised factors that drive download decisions [20, 25, 35] without being able to determine what happens once the app has been installed on the device. While some studies have relied on measurements taken on the handsets, they have focused on overall usage and how that is influenced by contextual factors [5, 14, 39], leaving commercial reports by mobile analytics companies the only source of information about what happens once the app has been downloaded. These reports suggest that usage dwindles significantly after installation (i.e., low retention), and losing even 75% of users after one day is common[6]. Typically, retention suffers the most on the first few days. Retention loss is slower after this. In the 2016 mobile customer retention report[6], 25% of users are lost on day 1, only about 12% are lost by day 7, and this drops to about 7% by day 30. However, to date, no independent research has verified these findings. In this section we present the first *independent* and large-scale study to investigate whether high retention loss in the first few days is indeed the case in the wild.

3.1 Retention Rate

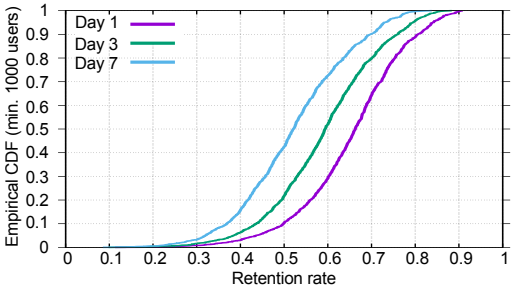
Retention rate on day d is defined as the percentage of users that continue using an application d days after first usage. To estimate retention rates, we identify for each user and application the first and last time the user launched the application. To ensure usage behaviour is correctly captured for retention rate of up to d, for each app we only consider those users who had not been using the app within d days of the last measurement day (July 14th 2015).

The retention rates of mobile apps in our dataset are illustrated in Figure 2a. From the figure we can observe that, while retention rates of many applications indeed are low, there are several apps with a healthier usage life-cycle. Overall, for all apps, the first day retention is as low as 36% with 7 day retention falling below 30%. However, our results also suggest that this effect is mainly due to many apps receiving only few users. Indeed, retention rates for apps with at least 10 users until the last measurement day show much healthier behaviour, with first day retention being close to 50%. For

[6] https://www.braze.com/blog/app-customer-retention-spring-2016-report/



(a) Retention rate largely depends on the initial number of users, with popular apps staying healthy for longer periods of time.



(b) Empirical cumulative distribution of retention rate for Apps with at least 1,000 users.

Fig. 2. Retention rates and their distribution.

apps with at least 1000 users the same figure rises to 62%. For the most popular 100 apps, first day retention is even as high as 68% and after 7 days the retention remains higher than 50%. In summary, our analysis supports the view that retention rates of apps tend to be low, with the usage witnessing a steep decline particularly after the first use. However, our analysis also calls into contention some of the claims made by analytics companies, indicating that the number of users mediates the retention of applications. We note that for apps with small user base (e.g., only $10-15$ users), some fraction of the retention could be potentially explained by developers of the apps continuing to test and use their app. Unfortunately, identifying these users from the Carat data is not possible.

To further shed light on retention patterns, Figure 2b illustrates the empirical cumulative distribution function of retention rates for apps with at least $1,000$ users. In the plot we separately consider the retention rates of day 1, day 3 and 7. These days were chosen for our results to be comparable with results published[1]. From the plot we can see that high retention rates are rare. Indeed, only $10-35\%$ of the apps have retention rates of 70%, and a mere 3% is able to achieve 75% retention rate on day 7. However, from the figure we also observe that extreme drops are rare, with less than 5% of applications having retention rates below 30%, i.e., the 80% drop reported by analytics companies is not common for apps that have been able to attract a sufficient user base (10 or more).

We observed similar patterns for apps with less than $1,000$ users. However, since there are orders of magnitude more apps with only a handful of users, as opposed to, for instance, hundreds of users, the retention rate of the entire data is then biased. Apps with 100 users or less are very volatile in terms of retention rate, since a drop of a single user already decreases retention by 1% or more. The plot for apps with at least 10 users within the whole measurement period follows a similar, but more jagged pattern, and rises much faster with retention rates around 10% lower than in Figure 2b.

This also further supports our earlier finding of retention being mediated by the size of the user base and confirms reports by others[2]. To verify this, we used Spearman correlation to assess the statistical dependency between usage counts and retention rates. To limit potential biases and noise in the retention rate estimates, we only considered apps with at least 10 users within the complete measurement period. The resulting analysis revealed the correlation to be statistically significant for all days ($d = 1, \rho = 0.199, p < .001; d = 3, \rho = 0.185, p < .001; d = 7, \rho = 0.165, p < 0.001$). For applications with higher usage count, correlations were slightly lower, but remained consistently significant.

We conclude that retention is indeed an indicator of the overall usage trend an application might experience over its lifetime. However, as we next demonstrate, it does not provide a full picture of

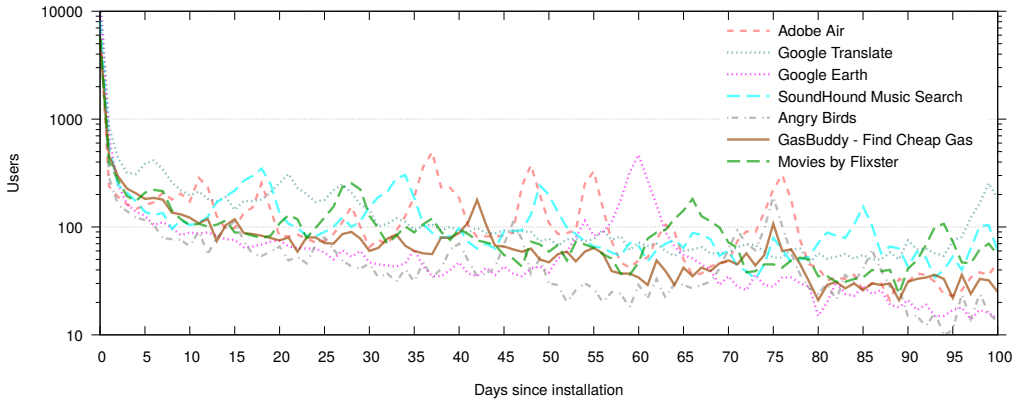


Fig. 3. App usage patterns do not always follow a simple falloff graph as suggested by retention rate.

app usage. In particular, we propose the use of trend filters that are able to compare apps regardless of their user base or absolute usage count, but instead to account for instantaneous trend patterns and seasonal effects.

3.2 Beyond Retention Rates

While retention is able to indicate the long-term usage trend of an app, it does not cover instantaneous popularity, trends or seasonal patterns. Furthermore, since app-usage is also conditioned on external context and occasion [5], apps with seasonal usage patterns, such as recommendation of lunch places, nearby gas stations or vacation-related are unfairly treated by retention rates. This is in particular true when the time window is short, such as the often cited 1-day/3-day/7-day retention characteristics. For instance, Figure 3 depicts usage patterns of exemplary applications from first day of usage until 100 days of usage. The selection of applications was done automatically using a peak detection algorithm that identifies significant peaks in usage after the initial slide in usage.

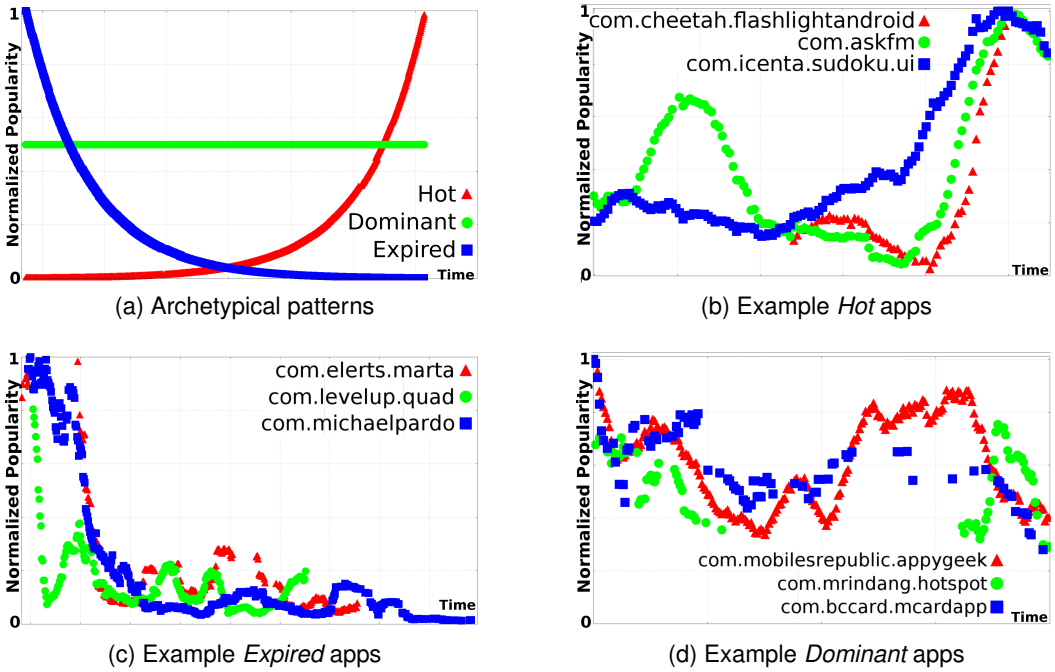
In the figure, *GasBuddy* is a representative example of an app with clear seasonal pattern. It compares fuel prices at nearby gas stations. The use of the application dwindles after day 1, but has recurrent spikes at biweekly and monthly intervals. Other examples include the *Adobe Air* game store/platform (regular peaks), or *Angry Birds* (many small peaks). Utilities, such as the music identifying *SoundHound*, *Google Translate*, and *Google Earth* (both irregular peaks) are used on-demand as their importance to users is situational.

To capture these effects and to provide a more accurate view of the usage of an application, in the next section, we propose a novel trend mining approach that captures application life-cycles and characterises the current stage of the app within its life-cycle.

3.3 Archetypical Trend Patterns

We focus on four archetypical trend patterns that are motivated by trend-decomposition models [32]. The choice of patterns is based on possible direction of the slope of the trend. The patterns *Hot* and *Expired* reflect rising and decreasing usage, whereas *Dominant* corresponds to only small change in constantly high popularity. The *Marginal*[7] pattern functions as a filter and covers those patterns that

[7]Apps with very low usage (*Marginal* pattern) closely resemble a straight line with constant value '1'. This is due to our pre-processing of applications to make them comparable: Normalisation with respect to the highest observed daily use followed by the interpolation of missing values (cf. Section 4).



(a) Archetypical patterns

(b) Example *Hot* apps

(c) Example *Expired* apps

(d) Example *Dominant* apps

Fig. 4. Archetypical trend patterns and example applications following these patterns.

are effectively non-existent as their usage features only few occasional individual usages on selected days (cf. Figure 4). Examples of their corresponding usage patterns are illustrated in Figure 4a. Figure 4b, 4c and 4d depict exemplary app usage evolutions for specific apps following one of these archetypical trend patterns.

4 MINING REAL-TIME TRENDS

We propose instantaneous application usage as a novel metric to accurately characterise the momentary popularity of mobile apps. Existing metrics, such as aggregated installation counts, co-installed apps and user reviews, often contain noise and biases making them an unreliable measure of an application's success or failure. Moreover, as shown in the previous section, these measures ignore effects of temporal and other seasonal or external factors.

Our trend filter abstracts away the user count or absolute usage, instead comparing applications solely based on their momentary trend potential. In this way, also apps with a small user base remain competitive in app recommendation systems.

4.1 Pre-processing and Data Preparation

The trend filter takes as input a usage time series $U(\mathcal{A}) = u_{\mathcal{A},1}, ..., u_{\mathcal{A},n}$ for any specific application $\mathcal{A}$. The usage data garnered from Carat sometimes contains missing values. To cope with these, as an initial pre-processing step, we fill in missing values $u_{\mathcal{A},j}$ using linear interpolation:

$$u_{\mathcal{A},j} = u_{\mathcal{A},i} + (j-i)\cdot\frac{u_{\mathcal{A},k} - u_{\mathcal{A},i}}{k-i}; i < j < k \in \mathbb{N}. \quad (1)$$

The trend filter then operates on the normalised absolute usage

$$\hat{U}(\mathcal{A}) = \hat{u}_{\mathcal{A},1}, \ldots, \hat{u}_{\mathcal{A},n} \quad (2)$$

$$\hat{u}_{\mathcal{A},i} = \frac{u_{\mathcal{A},i}}{\max_j(u_{\mathcal{A},j})}, i = 1..n \quad (3)$$

Note that while linear interpolation may not provide an accurate view of actual usage, it is sufficient for our trend filter since the key requirement is to preserve the sign of the usage time series' slope. In the following, we describe different trend operations that can be computed by our trend filter.

4.2 Trend Operation: Grouping Apps with Respect to a Specific Trend Pattern

To identify apps that follow specific archetype trend patterns (see Section 3.3 and Figure 4), a set of apps is clustered according to their similarity (Euclidean distance in the feature space) to these archetypes. Clusters are achieved with k-means clustering according to the input pattern's similarity to specific archetypical patterns. We utilise k-means since it allows to specify a fixed but arbitrary number of cluster heads. For instance, in our case we interpret the cluster heads as the four respective archetypical trend patterns *Hot, Expired, Dominant*, and *Marginal*. To compare normalised usage patterns via k-means, we measure similarity via their Euclidean distance in a feature space spanned by features capturing characteristics of the trend curve: *Area Under the Curve* (AUC), *Relative Peak location* (PEAK), *Slope* (SLOPE), and *Variance* (VAR) [8, 47]

$$\text{AUC}\left(\hat{U}(\mathcal{A})\right) = \sum_{i=1}^{n} \hat{u}_{\mathcal{A},i} \quad (4)$$

$$\text{PEAK}\left(\hat{U}(\mathcal{A})\right) = \arg\max_j \left(\hat{u}_{\mathcal{A},j}\right) \quad (5)$$

$$\text{SLOPE}\left(\hat{U}(\mathcal{A})\right) = \frac{\hat{u}_{\mathcal{A},n} - \hat{u}_{\mathcal{A},1}}{n-1} \quad (6)$$

$$\text{VAR}\left(\hat{U}(\mathcal{A})\right) = \frac{1}{n}\sum_{i=1}^{n}\left(\hat{u}_{\mathcal{A},i} - \hat{\mu}_{\mathcal{A}}\right)^2 \quad (7)$$

where $\hat{\mu}_{\mathcal{A}}$ describes the arithmetic mean of $\hat{U}(\mathcal{A})$.

The choice of these features is motivated by the nature of the trend patterns to distinguish. *PEAK* and *SLOPE* are able to distinguish *Hot* or *Expired* patterns, whereas the *AUC* distinguishes those from the other two constant patterns which have a much larger *AUC*. Finally, *VAR* and *AUC* are able to distinguish between *Dominant* and *Marginal* as the latter will have low *VAR* and high *AUC*.

Let N be the number of points, D the number of dimensions, and K the number of cluster heads. Based on the number of distance calculations, the time complexity of k-means is $\mathcal{O}(NKD)$. The space complexity of k-means clustering is then $\mathcal{O}\left(N(D+K)\right)$[21]. We note that variants of k-means exist with a tight asymptotic bound on the expected run-time complexity of $\mathcal{O}(\log K)$ [43].

4.3 Trend Operation: Extracting Representative Patterns

In order to identify a single representative usage trend pattern for a group of apps clustered to $\mathbf{C} = \{\hat{U}(\mathcal{A}), \ldots, \hat{U}(\mathcal{Z})\}$, this trend operation allows us to calculate a consensus $\hat{\mathcal{C}}(\mathbf{C})$ from all apps in the cluster. We exploit this trend operation, for instance, to identify a representative pattern of a Google Play category by computing the consensus for all apps in the category. It can also be applied to arbitrary groups of apps, for instance, to compare their average popularity (e.g. different groups of games or applications from a specific developer)

A challenge we address with this trend operation is that app life-cycles are not synchronised with respect to their absolute occurrence time. For instance, for the *Expired* pattern, the peak point for

each app in a respective cluster might be different and arbitrarily distributed over time. We therefore first synchronise all apps with respect to their peak as

$$\bar{u}_{\mathcal{A},l} = \hat{u}_{\mathcal{A},i} \quad (8)$$

$$l = i - \arg\max_{j} \left(\hat{u}_{\mathcal{A},j}\right) \quad (9)$$

From all usage timeseries $\hat{U}(\mathcal{A}), ..., \hat{U}(\mathcal{Z})$ in one cluster $\mathbf{C} = \{\hat{U}(\mathcal{A}), \ldots, \hat{U}(\mathcal{Z})\}$, a consensus timeseries $\mathcal{C}(\mathbf{C})$ is constructed as the mean over all time series in that cluster as

$$\mathcal{C}(\mathbf{C}) = \hat{c}_1, \ldots, \hat{c}_n \quad (10)$$

$$\hat{c}_i = \frac{\sum_{\hat{U}(\mathcal{I}) \in \mathbf{C}} \hat{u}_{\mathcal{I},i}}{|\mathbf{C}|}; i \in [1, n]. \quad (11)$$

Let $\hat{U}(\mathcal{I}) \leq \hat{U}(\mathcal{J}), \forall \hat{U}(\mathcal{J}) \in \mathbf{C}$ be the longest usage time series in the cluster $\mathbf{C}$ with $|\mathbf{C}| = m$ and $|\hat{U}(\mathcal{I})| = n$. Then, the time and space complexity of this trend operation is $\mathcal{O}(m \cdot n)$.

4.4 Trend Operation: Determining the App Life-cycle

Many applications have not completed their life-cycle instead reflecting an intermediate point of their respective life-cycle. For instance, an app might be in the beginning (initial stage), middle (rising) or end (past the peak) of a life-cycle. In order to find the stage of an app within a particular life-cycle, we apply alignment approaches [36]. In particular, the trend operation aligns representative trend patterns for archetypical life-cycles $L = l_1, \ldots, l_o$ to the normalised observed app usage history patterns $\hat{U}(\mathcal{A}) = \hat{u}_{\mathcal{A},1}, \ldots, \hat{u}_{\mathcal{A},n}$. The alignment found constitutes a sequence $\tilde{L} = \tilde{l}_i, \ldots, \tilde{l}_j$ that originates from L and is similar to the usage pattern $\hat{U}(\mathcal{A})$. $\tilde{L}$ possibly omits leading and trailing samples of L and may feature additional gap-symbols which are inserted via integer programming to minimise the difference between $\tilde{L}$ and $\hat{U}(\mathcal{A})$. In particular, a $n \times o$ cost matrix M, spanned by $\hat{U}(\mathcal{A})$ and L is generated by calculating all possible matchings between the l_i and $\hat{u}_{\mathcal{A},j}$ with respect to a distance cost function $c(l_i, \hat{u}_{\mathcal{A},j}) \rightarrow \mathbb{R}$ and a gap cost d:

$$M_{i,j} = \min\left(M_{i-1,j-1} + c(l_i, \hat{u}_{\mathcal{A},j}), M_{i-1,j} + d, M_{i,j-1} + d\right) \quad (12)$$

The M_{ij} constitute the minimum cost to align $l_1 \ldots l_i$ with $\hat{u}_{\mathcal{A},1} \ldots \hat{u}_{\mathcal{A},j}$. The optimal alignment $\tilde{L}$ is then found by traversing the minimum-cost path through M. Note that we set $M_{1j} = 0$ to allow $\tilde{L}$ to start at any position within L. Assuming a maximum sequence length of n, time and space complexity of this trend operation are $\mathcal{O}(n^2)$.

4.5 Trend Operation: Identifying Apps that Drive the Trend

Some groups of apps, for instance, specific Google Play categories, are dominated by individual highly popular apps. This may happen when users of popular apps try other, similar apps in the same category. Due to this, the observed usage of several apps in a category might be affected by individual popular apps so that multiple apps in that category rise or fall in popularity together. We are then interested in the usage trend normalised by the overall trend of the category or group of apps. In this way we are able to distinguish those apps, that drive and indeed exceed the category's trend performance from others that deviate with respect to the trend performance of the overall category.

This is achieved by normalising the performance of individual apps against the performance of the category. In particular, given a group of apps or category $\mathbf{C}$, we first compute the consensus $\mathcal{C}(\mathbf{C})$ of $\mathbf{C}$ (cf. Section 4.3). Then, for each app $\mathcal{A}$, we normalise its usage pattern $\hat{U}(\mathcal{A}) = \hat{u}_{\mathcal{A},1}, \ldots, \hat{u}_{\mathcal{A},n}$ with respect to $\mathcal{C}(\mathbf{C}) = \hat{c}_1, \ldots, \hat{c}_n$ as

$$\underline{\hat{u}_{\mathcal{A},i}} = \hat{u}_{\mathcal{A},i} - \hat{c}_i, i = 1..n \quad (13)$$

The resulting pattern $\underline{\hat{U}(\mathcal{A})}$ displays the performance of the app with respect to all other apps in the same category. A positive slope in $\underline{\hat{U}(\mathcal{A})}$ indicates that the app is performing better than its category while a negative slope indicates under-performance.

Assuming a total of $|\mathbf{C}| = m$ apps within the group of apps or category $\mathbf{C}$ with maximum pattern length $|\hat{U}(\mathcal{A})| = n; \mathcal{A} = \arg\max_{\mathcal{B}\epsilon\mathbf{C}} \left(|\hat{U}(\mathcal{B})|\right)$, the time complexity of this trend operation is $\mathcal{O}(n \cdot m + n + m)$. The space complexity is $\mathcal{O}(n \cdot m)$.

5 EMPIRICAL EVALUATION

To demonstrate the value of the trend filter, we carried out experiments using the Carat dataset (cf. Section 2). We first show that our trend filter is capable of identifying known success and failure stories. Afterwards, we analyse the accuracy of our approach for the prediction of the respective trends by computing the distance of the predicted archetypical trend pattern to the actual trend in usage data for that app. Next, we investigate the frequency in which trends occur within different application categories and finally, we discuss the impact individual apps can have on the usage performance of similar apps.

5.1 Validation with Well Known Apps

We begin our evaluation by demonstrating that the trend filter can correctly categorise the life-cycle state of apps. We consider eight apps whose usage patterns are well known and reported. Table 1 summarises the expectations for these apps. These apps also serve as representative examples of different types of trend patterns encountered in the data. Four of the eight apps demonstrated increasing popularity during the data collection period (June 2012 - July 2015), whereas the remaining four were popular early on but dwindled since. We stress that our aim is not to determine whether these apps were success or failure stories, but to highlight the stage of their popularity life-cycle.

Table 2 illustrates for each of these apps the distance to the nearest of the four representative archetypical patterns, and the corresponding trend prediction. In particular, this is the cluster to which the k-means groups these apps according to the Euclidean distance in the feature space between their observed usage pattern and the respective trend pattern. This Euclidean distance expresses the confidence of our trend filter on the respective categorisation. In each of the four dimensions *AUC, PEAK, SLOPE* and *VAR* (cf. Section 4.2), the Euclidean distance ranges from 0 to 1. Consequently, the overall Euclidean distance ranges from 0 to 2 with an average of 1.0. As further detailed in Section 5.2, we apply a confidence threshold of 0.4. For apps with a confidence value smaller than the confidence threshold, our trend filter predicts the respective trend. If the Euclidean distance to the closest trend exceeds the confidence threshold, no trend estimation is made.

From the apps with increasing popularity, *Evernote* is correctly predicted as *Hot* and *WhatsApp* is identified as *Dominant*. Among the apps with decreasing popularity, the trends of *Flappy Bird, Weibo* and *QQ* are identified as *Expired*.

For the remaining apps (*Vine, Snapchat, Path*), the trend filter has low confidence and therefore does not suggest a trend. The low confidence for assigning a trend to *Snapchat* can be attributed to the fact that the app only started to emerge during the data collection period, and was yet to experience an exponential rise in popularity. Therefore, even though the *Hot* archetypical pattern was the closest trend pattern found, it is not predicted as such due to its larger Euclidean distance of 0.4220 in the feature space. The low confidence for the *Path* app indicates that it is still in the transition between life-cycle states. As seen from Figure 5, the popularity of Path is at its peak, which explains the dominant pattern being the closest one. However, we can also see how the trend curve is starting to decrease, suggesting that Path would later enter the *Expired* state.

Table 1. Representative examples of popular apps whose changes in popularity patterns are well documented, and which were considered in our evaluation.

Apps expected to exceed expectation

Vine has been one of the hottest video sharing apps on Google Play between 2012 and 2016. It was particularly popular among young performers and musicians. While the app was withdrawn from the marketplaces in 2016, it was popular during the period of investigation considered in this paper (e.g., https://www.theverge.com/2016/10/28/13456208/why-vine-died-twitter-shutdown).

Evernote is an app that focuses on taking, sharing and managing notes. Compared to other social media apps, it has more professional reputation and rates 4.6 in Google Play and 4.5 in AppStore. The popularity of Evernote increased steadily during the data collection period until late 2015. Thus, Evernote mostly reflects the *Hot* archetype, but also reaches *Dominant* stage within the data collection period.

Snapchat is a chat application based on short video clips. The app is rated 3.9 in Google Play (3 in AppStore). Snapchat has steadily increased in popularity since 2012, starting to reach peak popularity in 2015. Snapchat initially does not have a clear trend pattern, only reaching *Hot* stage toward the end of the data collection period.

WhatsApp is currently the most popular messenger in over 100 countries (excluding USA and China) and the most popular worldwide (https://www.statista.com/statistics/258749/most–popular–global–mobile-messenger-apps/), allowing calls, group messages, and picture sharing. WhatsApp received high ratings, with an average score of 4.4 on Google Play and 4.5 on AppStore. Its consistent high popularity is representative for the *Dominant* trend.

Once popular apps, trailing expectations

Path is a social networking app that struggled to gain significant momentum until it was taken over in 2015 by the company behind Kakao Talk. The takeover briefly resulted in a surge of popularity, even if the app has since seen its popularity dwindle. Hence, Path is an example of an app that goes through multiple trend cycles, including *Dominant* and *Expired* within the data collection period.

FlappyBird is an arcade-style side-scroller game, which became a viral hit before taken down by the creator. After taken down, it's popularity quickly started to dwindle, making Flappy Bird a clear example of an app with an *Expired* trend pattern (see, e.g., https://mashable.com/ 2014/02/10/flappy-bird-story)

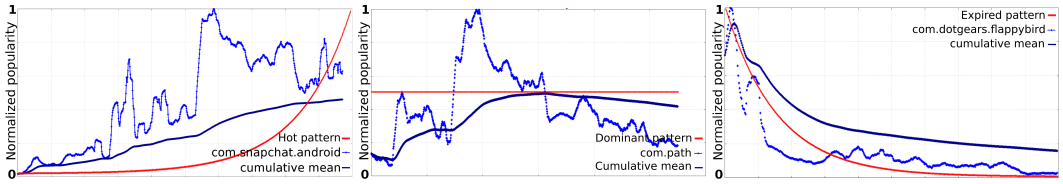
Weibo is a popular Chinese social media and microblogging app. It has been the most popular social media platform in China, but has since lost its position to WeChat. While the number of users for Weibo remains high, they are dwindling. It is a representative example of a popular application with an *Expired* trend.

QQ is a popular Chinese instant messenger. Similarly to Weibo, it has been among the most popular messaging apps in China, but has seen its popularity growth stall with WeChat becoming the most popular app. Hence, QQ is another example of an app that has reached the end of its popularity growth and is currently in the *Expired* state.

Table 2. Categorisation of example applications.

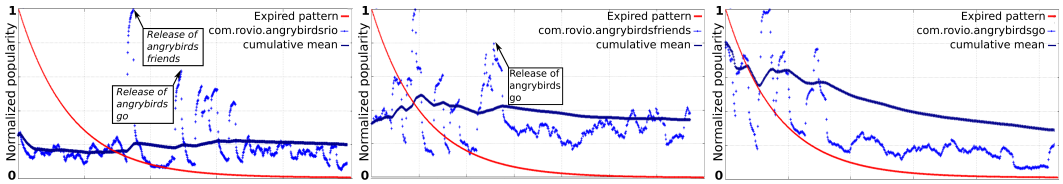
App	Category	Closest Pattern	Distance	Trend	Expected Trend
Vine	Entertainment	*Hot*	0.4220	–	*Hot*
Evernote	Productivity	*Hot*	0.3956	*Hot*	*Hot, Dominant*
Snapchat	Social	*Hot*	0.5399	–	Hot
WhatsApp	Communication	*Dominant*	0.1186	*Dominant*	*Dominant*
Path	Social	*Dominant*	0.4467	–	Dominant, Expired
Flappy Bird	Game - Arcade	*Expired*	0.0575	*Expired*	*Expired*
Weibo	Social	*Expired*	0.2343	*Expired*	*Dominant, Expired*
QQ	Social	*Expired*	0.2854	*Expired*	*Dominant, Expired*

We stress that the *Expired* pattern should not be interpreted as overly negative. On the contrary, it indicates that the app was successful in gathering a significant usage, but has experienced significant loss in usage thereafter. As discussed in Section 3.1, it is a natural matter of retention that the usage drain is significant, and *Expired* simply means that the app has surpassed its popularity peak. Consider, for instance, the Angry Birds series of apps. As indicated in Figure 6, the patterns of most Angry Birds apps closely resemble the *Expired* pattern (shifted by their respective release date) even if most of them can be considered to be exceptionally successful. However, we can also observe from the figure that apps in the same theme or product family have potential to benefit from each other as a compounding effect. For instance, the popularity of older Angry Birds apps increases when a new version is released, so that the apps re-enter the *Hot* state for a period of time. Indeed, we see in



(a) Trend evolution of Snapchat (b) Trend evolution of Path (c) Trend evolution of Flappybird

Fig. 5. Trend evolution, cumulative mean closest archetype trend pattern for example apps. The timeline is normalised for comparison between the first and last usage data for each respective app in our dataset (Snapchat: November 15th, 2012 – July 13th, 2015; Path: July 7th, 2012 – July 13th, 2015; Flappybird: February 10th, 2014 – July 13th, 2015).



(a) Trend of Angry Birds Rio (b) Trend of Angry Birds Friends (c) Trend of Angry Birds Go

Fig. 6. Trend evolution, cumulative mean closest archetype trend pattern for example Angry Birds apps correlated in their usage performance. The timeline is normalised for comparison between the first and last usage data for each respective app in our dataset (Angry Birds Rio: July 7th, 2012 – July 12th, 2015; Angry Birds Friends: May 21st, 2013 – July 13th, 2015; Angry Birds Go: December 28th, 2013 – July 12th, 2015).

Figure 6a and Figure 6b how the much older *Angry Birds Rio* also experiences a rise in popularity at the time the *Angry Birds Friends* and *Angry Birds Go* are released. These peaks can also be observed at very similar times for the original *Angry Birds* and the older *Angry Birds Seasons*.

5.2 Evaluation of the Quality of the Trend Prediction

We next analysed the quality of the trend prediction with regard to the distance to the respective trend pattern (Figure 4a). Table 3 summarises the frequency of the four trend patterns in exemplary Google Play categories.

Of the apps associated with one of the four archetypical trend patterns, less than 0.1% gather a constantly high user base (*Dominant* apps). Fewer than 1% are *Expired*, and apps associated to the *Hot* category account for about 2 to 7% of all relevant apps (6% across all categories). The mean Euclidean distance of all individual apps to the associated (i.e. closest) trend pattern is in the order of 0.95, with a variance of 0.2 as detailed in the table.

These results are conditioned on the confidence associated with such prediction. With low confidence, the algorithm does not predict a trend for a usage pattern but only predicts a particular trend pattern, when it has the closest Euclidean distance of all trend patterns and when this distance to the respective cluster centroid (*Hot*, *Expired*, *Marginal*, or *Dominant*) falls below 0.4.

Table 3. Percentage of *Marginal*, *Expired*, *Dominant*, and *Hot* apps for 17 exemplary categories.

	Books and Reference	Business	Comics	Communication	Education	Entertainment	Family	Finance	Game (Action, Adventure, Arcade, Board)	Health and Fitness	Lifestyle	Media and Video	News and Magazines	Personalisation	Productivity	Tools	Travel and Local
Marginal apps(%)	43.28	38.16	31.94	41.2	38.78	38.63	49.81	47.2	38.62	42.54	40.3	46.1	41.35	40.15	44.78	43.97	39.62
From the rest:																	
Hot apps(%)	1.33	2.65	4	7.44	.81	2.33	4.65	4.4	1.73	3.92	3.17	6.5	4.74	2.25	6.78	5.98	2.95
Dominant apps(%)	.04	.06	0	.14	0	.03	0	.03	0	0	.07	.11	.07	.02	.06	.07	0
Expired apps(%)	.25	.74	.8	1.91	.03	.52	.9	.63	.52	.36	.37	0	.95	.74	2	1.51	.64
Mean Eucl. Dist.	.974	.967	.914	.906	.996	.958	.921	.961	.938	.967	.957	.947	.926	.961	.926	.925	.958
Variance	.010	.012	.018	.027	.008	.013	.020	.016	.016	.013	.014	.018	.023	.015	.023	.024	.012

The value of 0.4 was chosen empirically as (1) a sphere of radius 0.4 covers approximately[8] 10% of the total volume in a 4D unit space (spanned by the four feature values, each in [0,1]) and (2) since it was able to clearly separate the apps belonging to one of the four patterns from those that do not belong to it (cf. Figure 7). Note that it is challenging to define an absolute threshold, as the boundary between apps that still belong to one specific trend pattern and those that do not is floating.

In the figure, we have calculated the mean Euclidean distance for groups of apps clustered to the same of the four respective trend patterns to their nearest cluster centroid. The figure separates apps with a Euclidean distance greater than 0.4 from those with a smaller Euclidean distance.

For the apps associated with one of the trend patterns (i.e. Euclidean distance smaller than 0.4), the *Marginal* apps are most similar to their respective trend pattern and the mean Euclidean distance for *Hot*, *Dominant*, and *Expired* apps is for all categories sharply concentrated around 0.3. As we have seen in Table 3, the distance of the remaining apps to their nearest trend pattern is significantly higher. This means that they might follow other, more random usage evolution or experience constant fluctuation. For example, the application may be in an early stage of its life-cycle, or transitioning between trend patterns. While we focus in this study on *Hot*, *Dominant*, *Expired*, and *Marginal* patterns, our approach can be generalised to other types of patterns in the data. See also Section 8 for a discussion on further pattern types.

5.3 Distribution of Distinct Trend Patterns

We have investigated the count and frequency of diverse patterns in the Carat data. In particular, we are concerned with the number of different relevant trend patterns that can be found since the *Hot, Dominant, Expired*, and *Marginal* patterns constitute only part of the patterns present. To better understand this, we clustered the life-cycle patterns found for all apps in the carat dataset with k-means clustering and larger values of k. In all cases, as discussed in Section 5.2 and visible from Table 3, an overwhelming share of apps follow the *Marginal* cluster. For the remaining clusters observed, we found that few prominent clusters dominate. For instance, Table 4 displays for $k = 20$ the sorted arithmetic mean of the non-*Marginal* clusters[9] found among the apps in the carat dataset. In particular, Figure 8 depicts the distribution of mean cluster sizes we found after 10 runs of k-means with $k = 20$.[10] The *Marginal* pattern was constantly found by about one order of magnitude more often than all other patterns (not shown in the figure). The *Hot*, *Expired*, and *Dominant* trend have

[8] A more exact approximation would be 0.377, which we rounded up to 0.4 to achieve a slightly larger noise tolerance. Results achieved for 0.4 and 0.377 are nearly identical

[9] The *Marginal* cluster is not included in the table as it is uninformative due to the small number of samples for each of the *Marginal* patterns.

[10] $k = 20$ was chosen empirically by gradually increasing k until no clusters with significantly new patterns were found.

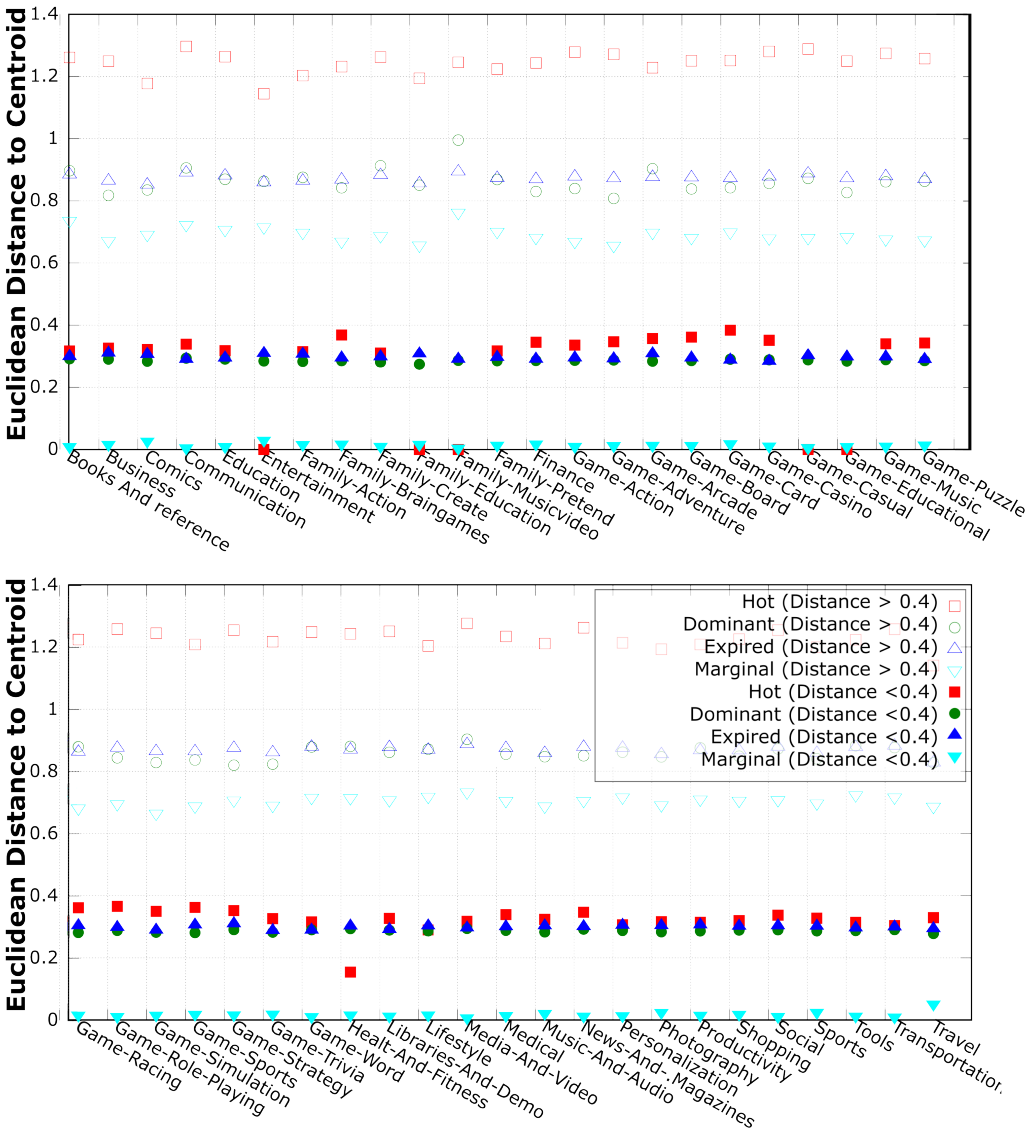


Fig. 7. Mean Euclidean distance to the respective cluster centroid (*Hot*, *Dominant*, *Expired*, *Marginal*) for apps associated with the respective trend pattern (low Euclidean distance) compared to those not associated with it (high Euclidean distance).

Table 4. Distribution of the arithmetic mean of clusters found (rounded to 1000 for space constraints).

C	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
$\overline{\mathbf{C}}$ $[10^3]$	629	820	952	1079	1163	1285	1342	1385	1474	1550	1707	1857	2023	2180	2302	2608	3159	3566	4482

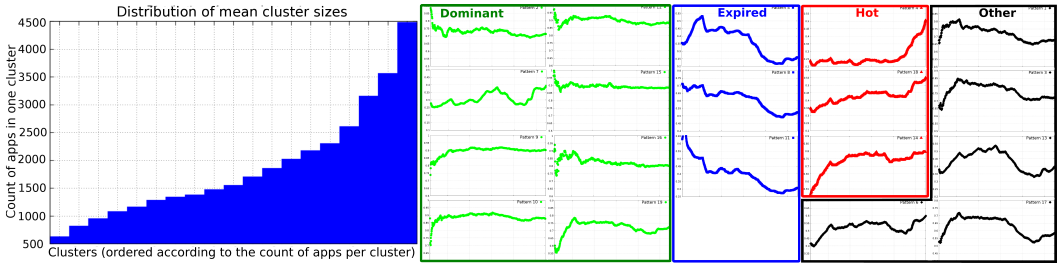


Fig. 8. Distribution of cluster sizes and consensus of the clusters found grouped with respect to their closest archetypical pattern.

been prominently found among the clusters computed in all runs of the k-means algorithm. Moreover, we can observe many of the smaller clusters to correspond to slightly modified variants of the four archetypes considered in our work. Note from the figure, that the three largest clusters jointly represent about a third of the apps with a long tail of small clusters. This shows that, although more than the *Hot*, *Expired*, and *Dominant* (and *Marginal*) patterns can be found in the data, the relevance of these four clusters is significant.

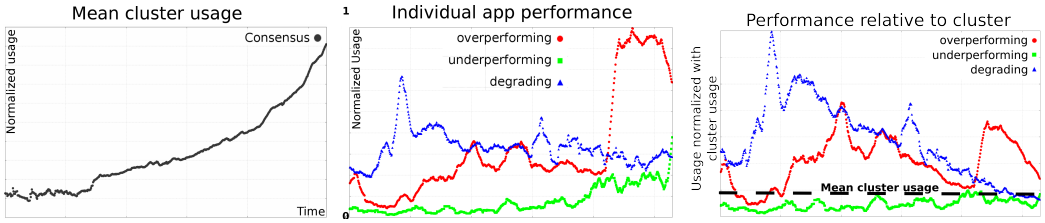
5.4 Impact of Popular Apps within a Category

We are further interested in the popularity of specific apps with respect to others. However, apps can fall into various categories and might not be comparable, such as, for instance, categories *Game* and *Business*. In particular, in the Google Play Store, some categories might increase in popularity while others decrease. As detailed in Section 4, such overall trend of the category might affect other apps within the category, so that an unbiased comparison across categories requires prior normalisation (cf. Section 4.5).

Highly successful apps have a significant impact on the trend of their category[11]. For instance, the Facebook mobile app dominates its category. Compared to the consensus of all apps in this category, *com.facebook.katana* features with 0.2105 the smallest Euclidean distance in the feature space to that categories' consensus pattern. Smaller apps might then appear to follow a rising trend while in reality they have merely been benefiting from the overall popularity of the category. For a fair comparison of apps that belong to different categories, the overall trend of the categories should therefore be subtracted.

For recommendation or analysis purposes, apps that drive the trend are especially interesting over copycat-apps which are worse than the trend. To objectively measure the performance of an app, free of the influence of its category, we calculate its performance relative to the performance of its category (cf. Section 4.5). To illustrate that normalisation against the overall trend in a category is beneficial to identify the actual instantaneous performance of an app, consider Figure 9 for three exemplary apps in one category. Figure 9a plots the consensus of the app category. Looking at an individual application's performance (Figure 9b), their trend is hardly visible. However, after normalisation with the cluster's consensus pattern in Figure 9c, the app represented with ● is over-performing as it regularly scores above the cluster's performance while the app labelled ■ is constantly under-performing. Finally, the third app (labelled ▲) is dominating in the beginning while then constantly decreasing in popularity. Observe that, in contrast to this normalised evolution, considering the apps individually, as in Figure 9b, the under-performing app actually appears to be rising in popularity while the app with decreasing popularity appears to remain stable.

[11] e.g. users trying out other similar apps and thereby boosting the popularity of copycat apps.



(a) Consensus of the cluster, describing the average popularity of apps in this cluster

(b) Normalised performance of three individual apps from this cluster

(c) Relative performance of the apps with respect to the cluster

Fig. 9. Performance of individual apps relative to the performance in their cluster.

6 PRACTICAL USAGE OF APP TRENDS

To demonstrate the practical value of our work, we now consider how trend information can affect app recommendations. We have implemented AppJoy [48] as a representative example of current state-of-the-art mobile app recommenders, and compared the recommendations provided by AppJoy against respective trend patterns of the recommended apps. AppJoy operates on so-called usage scores, which are constructed by aggregating (i) the time elapsed since the last interaction with an app (v_R), (ii) the number of times the user interacted with an app (v_F), and (iii) total duration of interaction time (v_D). Accordingly, AppJoy bases its recommendations on information sources that correspond to metrics which are used by handset-based mobile analytics tools, such as Google Mobile Analytics[12] and Countly[13].

AppJoy uses a prediction model that compares a user $\mathfrak{U}$'s profile to other users with similar application usage history. Let $\mathbf{S}(\mathfrak{U})$ be the set of applications used by $\mathfrak{U}$. Given an application $\mathcal{A}$ and $\mathfrak{U}$, we define $\mathbf{R}_{\mathfrak{U},\mathcal{B}}$ as the set of relevant applications $\mathcal{B}$ used by other users together with $\mathcal{A}$, so that

$$\mathbf{R}_{\mathfrak{U},\mathcal{B}} = \{\mathcal{A} | \mathcal{A} \in \mathbf{S}(\mathfrak{U}), \mathcal{B} \notin \mathbf{S}(\mathfrak{U}), size(\mathbf{S}_{\mathcal{A},\mathcal{B}}) > 0\} \tag{14}$$

where $\mathbf{S}_{\mathcal{A},\mathcal{B}}$ is the set of users who have used both $\mathcal{A}$ and $\mathcal{B}$. The relevance or occurrence probability of $\mathcal{B}$ for $\mathfrak{U}$ is then given by:

$$P(\mathfrak{U}_{\mathcal{B}}) = \frac{1}{size(\mathbf{R}_{\mathfrak{U},\mathcal{B}})} \sum_{\mathcal{A} \in \mathbf{R}_{\mathfrak{U},\mathcal{B}}} (dev_{\mathcal{A},\mathcal{B}} + \mathfrak{U}_{\mathcal{A}}). \tag{15}$$

In the above equation, $dev_{\mathcal{A},\mathcal{B}}$ denotes the average of the usage scores between users who have used both $\mathcal{A}$ and $\mathcal{B}$:

$$dev_{\mathcal{A},\mathcal{B}} = \sum_{\mathfrak{O} \in \mathbf{S}_{\mathcal{A},\mathcal{B}}} \frac{v_{\mathfrak{O} \vdash \mathcal{B}} - v_{\mathfrak{O} \vdash \mathcal{A}}}{size(\mathbf{S}_{\mathcal{A},\mathcal{B}})}. \tag{16}$$

Here, $v_{\mathfrak{O} \vdash \mathcal{B}}$ defines the usage score for application $\mathcal{B}$ used by user $\mathfrak{O}$:

$$v_{\mathfrak{O} \vdash \mathcal{B}} = \omega_R v_R + \omega_F v_F + \omega_D v_D, \tag{17}$$

weighted by ω_R, ω_F and ω_D. Given $P(\mathfrak{U}_{\mathcal{B}})$, AppJoy returns the apps with highest score as recommendation Φ.

To illustrate the value of trend and life-cycle information, we ran the AppJoy recommender and our trend analysis for a subset of the data containing 4,500 users and 1,000 most frequently used applications in the dataset. As our test period we selected October 2014, because it does not

[12] http://www.google.com/analytics/mobile
[13] http://www.count.ly

Table 5. Statistics of the best 20 recommendations for the top 1000 applications during October 2014. Apps predicted as *Hot* and *Expired* closely resemble the archetypes described in Figure 4a.

Week	Rec. Hot	Rec. Expired	Total Hot	Total Expired	Div.	Nov.	Acc.	Div. w/o expired	Nov. w/o expired	Acc. w/o expired
1	8	5	219	163	-	-	0.02	-	-	0.02
2	7	6	229	158	0.80	0.98	0.03	0.90	0.90	0.12
3	8	7	232	154	0.62	0.81	0	0.54	0.73	0.10
4	10	9	225	150	0.56	0.75	0.11	0.50	0.68	0.11

feature any major European or international events that might cause seasonal fluctuation, e.g., mobile Christmas calendars or Eurovision song contest voting apps. As training data we selected all data accumulated between January 2014 and September 2014. Given the test data, we used AppJoy to generate recommendations in an incremental fashion for each week. In particular, we generated recommendations for the first week, then included the data from this period in the training data and generated recommendations for the second week, and so on. We also generated trends for every week, taking into account the life-cycles of the past year, starting from January 1st 2014. We counted (i) how many recommended applications are grouped as *Expired* or *Hot* apps, and (ii) how these compare with the total number of *Expired* and *Hot* apps in the top 1000 applications. We also calculated temporal diversity, novelty, and accuracy for the recommendation lists [27]. Diversity represents how the recommendations change over time, whereas novelty describes how many new recommendations are seen compared to the later ones. Novelty of the recommendations relates closely to the trends, because changes in trends should affect new recommendations. Given two sets $\mathbf{A}$ and $\mathbf{B}$ of apps and the set Φ of all recommended apps, as well as depth N, these metrics are defined as

$$diversity(\mathbf{A}, \mathbf{B}, N) = \frac{|\mathbf{B} \setminus \mathbf{A}|}{N} \tag{18}$$

$$novelty(\mathbf{A}, N) = \frac{|\mathbf{A} \setminus \Phi|}{N} \tag{19}$$

$$accuracy(\mathbf{A}, \Phi) = \frac{size(\mathbf{A} \cap \Phi)}{size(\Phi)}. \tag{20}$$

Results of our analysis are shown in Table 5 for the top-20 recommendations given to all users. The results indicate that the number of *Hot* apps recommended for each week is small and comparable to the number of *Expired* apps recommended in the same time. Given that we have generated in total $90,000$ recommendations for $4,500$ users each week, the amount of *Hot* recommended corresponds to a very small percentage of the entire set of recommendations. Within the top 1000 apps, more than 200 applications each week can be classified as *Hot*, and about 160 applications as *Expired*. On average, only 3.6% *Hot* apps are recommended, compared to 4.3% *Expired* apps. When *Expired* apps are removed from the recommendations, both novelty and diversity decrease, but accuracy increases slightly. The main reason for this behaviour is that the metrics used by AppJoy to generate recommendations require sufficient amount of usage before an app is recommended. However, once sufficient usage has been observed, the app can already be past its "best before" date as the recommendation model does not separate between *Hot* and *Expired* apps. Integrating usage trend information as part of the recommendation process can help to overcome this issue and improve the overall quality of recommendations.

Our results highlight how AppJoy requires apps to have sufficient popularity before they are recommended, and that both trend and dynamics of app usage are disregarded in the recommendation process. Indeed, apps tend to remain in the recommendation list until another app reaches the same level of popularity. As such, AppJoy suffers from poor recommendation diversity, and thus further

enforces the Matthew effect that popular apps are becoming more popular at the expense of new, emerging, and potentially interesting apps.

7 RELATED WORK

In this section we discuss recent advances in trend mining and trend detection as well as mobile app recommendation systems and how our proposed technique exceeds this state-of-the art.

Trend Prediction in numerical timeseries data is an important and well studied field in timeseries forecasting [19]. A time series is a series of, often real-valued numbers that occur over a discrete time. The analysis of timeseries traditionally assumes a time-invariant generation function so that the timeseries shows a stationary behaviour. To arrive at such behaviour, a typical approach is to detect and remove trend and seasonal components from the timeseries [7]. Typical methods stem, for instance, from linear regression in order to model linear or polynomial trend behaviour [9]. In addition, also statistical methods, such as moving average or splines techniques are employed [1].

However, the performance of prediction approaches (measured by the RMSE), does not translate into improvement in accuracy for the top-N task [13, 40]. In addition, and in contrast to these approaches for predicting app performance, the assumption of an underlying time-invariant generation function does not hold in our case as app popularity is conditioned on external factors and is also subject to aging. In addition, trend in timeseries analysis describes a linear, polynomial or exponential behaviour whereas we are interested to describe the trend as a life-cycle of an app, potentially also reflecting past behaviour, such as a steep rise followed by a drastic loss in popularity (*Expired* pattern). Therefore, we instead propose VAR, AUC, PEAK and SLOPE features to describe the shape of a usage or trend pattern.

Commercial **Trend Mining** systems include Google Trends[14], which monitors the frequency of words in search queries related to real-world events, and the trending topics list of Twitter, which uses the frequency of hashtags and noun expressions to determine popular topics. Related academic works include detecting emerging trends in real-time from Twitter [4, 10, 31], mining of news discussions or other text documents for trends [37, 41], and analysis of web behaviour dynamics [38], which are all indirect and subjective measures that might suffer from fraud. Our work is capable of operating solely on app usage information whereas these works operate on co-frequency patterns between words or n-grams.

By exploiting actual usage (in contrast to downloads, likes, ratings, and similar measures), our approach can identify and compare trends of apps regardless of their absolute user count, downloads, or installations. In this way, well-known apps can be compared to less well-known newcomers, and may show an inferior trend performance. Hence, exploiting our trend filter, promising future stars are potentially spotted earlier.

Mobile App Recommendation systems utilise a multitude of features to rate the relevance or popularity of a respective app. Among these, user reviews are a prominent source for app recommendation systems [15]. However, empirical studies have shown that reviews typically contain several topics, which are seldom reflected by the overall rating [24, 34]. Motivated by these studies, several works on using sentiment analysis and summarization techniques for mining app reviews have been proposed. Chen et al. [12] identify reviews that are most informative to developers, whereas Guzman and Maalej [18] use sentiment analysis and language processing to extract user opinions for different features in a mobile app. Recommendations, however, are in general prone to fraud and are inaccurate and noisy as they are based on free-form textual descriptions. Our trend filter is not affected by such subjective and biased information as recommendations, since it is conditioned on actual usage trend.

[14] google.com/trends

Jovian et al. [29] point out that version information should further impact the recommendation score as the popularity of an app might be affected by the change in version. We remark that the function of popularity conditioned on the version number is not necessarily monotonic or even increasing. Our trend detection approach, however, is able to detect such changes implicitly whenever alterations in the usage trend result from a version update. In addition, Lim et al. [28] point out that user behaviour is country specific, so that recommendation systems should adapt to such properties. Our proposed trend filter is also able to filter usage trends in a specific population, such as geospatial, age, or gender.

In order to improve the above mentioned global solutions in which recommendations are identical for all users, individual preferences are considered. Peifeng et al. [49] argue that an app recommendation system has to take into account also the set of already installed apps. They compare a 'tempting' value of a new application to a 'satisfactory' value of already installed applications of the same type. Another example is AppJoy [48], which employs item-based collaborative filtering to recommend apps based on personalised usage patterns. AppBrain[15] compares recommendations within the same category by monitoring the installation history of apps. Also, AppAware [16] provides recommendations by integrating the context information of mobile devices. Such personalised recommendations are also possible exploiting our trend filter by applying it to a personalised subset of apps. Moreover, users often use several apps within the same category [50] and the overall usage session times tend to be short, and depend on a wide range of contextual factors [5, 14]. Other approaches consider, for instance, the user's privacy expectations on a given app-type for recommendation [30]. In addition, co-usage of apps can be exploited for app recommendation as detailed in [44]. Responding to this observation, the AppTrends approach was proposed to base the recommendation on frequency of co-usage of apps [2]. In contrast to our work, AppTrends does not exploit usage trends of individual apps but instead co-usage with other installed apps. Our trend filter could be applied in addition to further improve app recommendations. Indeed, combinations of these individual recommendation systems to form multi-objective app recommendations have the potential to further improve accuracy in the recommendations [46].

Also, Petsas et al. [35] demonstrated that user preferences tend to be highly clustered and following various trends over time, with users showing interest in a small set of app categories at a time. Our work complements existing solutions by providing mechanisms for analysing and understanding application usage relative to the dynamics of the app's instantaneous popularity in a marketplace.

Temporal Recommendations in app recommender systems has been addressed in the literature to overcome the problem that existing recommender systems merely recommend apps that users have experienced (rated, commented, or downloaded) since this type of information indicates user preference. As a result, apps which are relevant but never experienced by users are not featured in the recommendation lists. To mitigate this problem, Bhandari et al. propose to recommend serendipitous apps using graph-based techniques [52]. While this approach is able to avoid over-personalisation, the recommendations are not correlated with popularity of apps since usage count is disregarded. Another approach is to utilise usage behaviour in apps to impact temporal recommendations, as e.g. suggested by Zhung-Xun et al. [53]. In contrast to our approach, the authors investigate individual usage traces to build up personalised recommendations. As described in their work, this approach necessarily suffers from retention effects as all usage patterns have a tendency to decline and furthermore, since the sample points are small because only a single users data is considered. Consequently, this approach suffers from both these biases. This shows that temporal recommendation is a challenging task and only access to usage behaviour of a large crowd of users mitigates the described biases. Our approach is a potential tool to tackle these challenges. Only when the crowd of users providing usage

[15] http://www.appbrain.com

information is sufficiently large, biases due to individual behaviour and noise diminish and crowd usage behaviour surfaces.

Some commercial app analytics tools, such as Google Mobile Analytics and Countly's Mobile Analytics, follow a similar path by focusing on monitoring statistics of individual apps, covering information about usage session frequencies, lengths of usage sessions, extent of in-app purchases, and so forth. In contrast to our work, these solutions do not consider popularity of apps conditioned on whether they follow specific trend patterns. The importance of app popularity has also recently been reported [51]. The authors proposed a HMM-structure in order to model and predict app popularity. In contrast to our work, the authors exploit temporal observations of rankings, user ratings and user reviews, rather than actual app-usage statistics and thereby directly accesses app popularity.

8 DISCUSSION

Representativeness of Carat Data Our analysis considered data collected through the Carat application, which has been originally designed for energy management. As a result, the sample population is likely biased towards people who either have battery issues or are interested in monitoring the performance of their device. This may result in our retention estimates being lower for some applications than what they would be for the entire smartphone user community. Our results thus should be interpreted as conservative lower bounds, particularly for apps with high energy consumption. Compared to reports by commercial analytics companies, our retention estimates were generally slightly higher, suggesting that retention patterns may not be as dire as analytics companies would lead to believe. Given the scale (339, 842 users and 213, 667 apps) and duration (approx. 36 months) of our analysis, the results in this paper serve as starting point for obtaining a better understanding of mobile app usage patterns in the wild.

Trend Archetypes We considered four trend archetypes, which were motivated by the most common slopes of the popularity graph (increasing, decreasing, constant with high popularity, and constant with low popularity). While these are the most prominent patterns in our data, also other types of patterns are possible. For example, popularity increases seen for older versions of Angry Birds apps whenever a new version is launched (see Fig. 6). Note that many of these patterns can be captured with a short-term variant of our life-cycle model. For example, the popularity boosts correspond to a model where an app re-enters the *Hot* phase temporarily, before progressing into *Expired* states. A modified model focusing on shorter duration trend patterns can extend the prediction to better cover also shorter-term fluctuation and temporal trend behaviour.

Temporal Recommendations and Trends Analysis of recommendations produced by AppJoy indicated that these do not reflect dynamics in actual application usage. Other state-of-the-art mobile app recommenders, such as Djinn [23] and GetJar [40], also include usage data, but do not model this as trend information. In particular, Djinn considers triples of item (e.g. app), usage, and user, but does not consider the time dimension, which is important to extract trend information. Similarly, although GetJar recorded the number of days the app was used (similar to retention rate), the time dimension was otherwise ignored as binary daily usage information is not combined to app-trends. To facilitate users to discover up and coming applications, and to help them avoid apps that are long past their popularity peak, the trend information could be integrated as part of the recommendation process, for instance, by considering it as part of the usage scores used by AppJoy or considering more complex dynamics models, for example, as part of latent factor models [26].

Application Potential We have demonstrated the benefits of considering app trend information for mobile analytics and app recommender systems. Another use for trend information is providing developers early feedback about the current popularity of their applications, which they can then

use to take countermeasures against negative popularity fluctuations. Trend state can further be correlated with other factors, such as usability gathered through interaction metrics [39], to provide more detailed feedback of the possible reasons in popularity fluctuations.

Beyond providing app developers with tools to understand the state of their app, trend can also be used for marketplace analytics to support advertising strategies. On the device side, trend status can be used as an additional metric to identify the most redundant applications for removal to reduce clutter on the user interface. The app-filter also enables detecting apps that are rapidly gaining in popularity, which could be used, for instance, by in-app advertisers to entice new app developers as customers or for dynamic pricing models.

Application trends are also potentially a powerful source of information for characterising and understanding user interactions, and trend information can be used to support user modelling. For example, users with consistently many *Hot* applications are continually shifting their application usage, whereas those with many *Expired* or *Marginal* apps are likely to remain faithful to the apps they originally chose.

Android vs iPhone The Carat app is available for Android and iPhone. In this work we focus on Android devices due to their application naming policy described previously in this work. With more controlled access to the app market, iPhone population could potentially demonstrate different retention patterns, trends, and trend patterns.

Implications The analysis has shed light on the complexity of application usage patterns and suggests that the trend life-cycle stage of an app together with its retention should be considered when analysing success and failure of mobile apps. For example, apps that currently have a large number of users but already show signs of losing that popularity can be recognised and classified earlier. Trend information can also be given to developers to help them reacting to changes, for example, by adding new functionality into the app when the popularity is starting to dwindle. Similarly, app recommendation systems could take trend information into account, for example, by highlighting some applications in the *Hot* stage and assigning less weight to apps in the *Expired* state. This could potentially increase diversity and serendipity of the recommendations without affecting their relevance.

9 SUMMARY AND CONCLUSION

We have presented the first ever independent study of retention rates in the wild. Our analysis shows that, on average, applications lose 65% of their users in the first week, but the effect is mediated by overall user count as applications with over $1,000$ users show much higher retention rates. We also demonstrated that, contrary to reports in the literature, severe losses in usage are rare, with less than 10% of apps losing over 80% of users in the first week. We demonstrate that retention rates are an insufficient metric of an application's success as they ignore effects of seasonality and external factors. In particular, we demonstrated that applications follow different trend patterns which are not captured by retention.

As second contribution, we proposed a novel app-filter that categorises applications according to their currently followed usage trend. We focused on four characteristic trends: *Marginal* apps with only few users, *Dominant* applications of permanent high popularity, *Hot* apps with rapidly increasing popularity, and *Expired* apps which experience a drastic drop of the usage. We observed that about 40% of the apps are *Marginal*. We analysed application categories from Google Play and show that, for example, during the year 2014, 7.5% of communication apps have been *Hot*, only 0.1% were *Dominant*, and almost 2% were *Expired* apps. This kind of analysis can, in the future, lead application developers to follow needs and desires of the users in faster pace.

As a practical use case of our work, we considered how our trend-filter can benefit mobile app recommenders by enabling recommendations to focus on those apps that are rising in popularity. Trend pattern analysis can be used to strengthen existing heuristics such as interaction rate, download counts, and reviews, and even give a more direct way to produce in-the-wild recommendations taking into account the usage history and trend-pattern of the application. Our analysis shows that only 3.6% of the recommendations are for apps which are currently rising in popularity, and that overall recommendations have low novelty and temporal diversity. We also demonstrate that the accuracy of the recommendations can be improved by considering trend information.

Another prominent issue in recommender systems is the cold start problem when insufficient data has been collected on a new user. The trend filter is not affected by this problem with regard to new users as usage trends are build by other users. However, we remark that for a new application, trend estimation is volatile over the first days as usage data has to be collected first. Although exploiting trend filters, apps of smaller user base are empowered to content with highly popular apps, note that the potential risk of recommending badly maintained apps is low as such apps likely boast a less satisfied user population and hence feature an inferior usage trend pattern compared to well maintained apps with a satisfied user population. Summarising, using our proposed trend filter has the potential to increase the success probability in finding good and reliable apps and that poorly maintained apps will not feature a positive trend for long.

10 ACKNOWLEDGEMENTS

This work was in part supported by the Academy of Finland grants 319017 and 297741.

REFERENCES

[1] Hirotugu Akaike. 1974. A new look at the statistical model identification. *IEEE transactions on automatic control* 19, 6 (1974), 716–723.

[2] Donghwan Bae, Keejun Han, Juneyoung Park, and Mun Y Yi. 2015. AppTrends: A graph-based mobile app recommendation system using usage history. In *Big Data and Smart Computing (BigComp), International Conference on*. IEEE, 210–216.

[3] Ricardo Baeza-Yates, Di Jiang, Fabrizio Silvestri, and Beverly Harrison. 2015. Predicting The Next App That You Are Going To Use. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining (WSDM '15)*. ACM, 285–294.

[4] James Benhardus and Jugal Kalita. 2013. Streaming trend detection in Twitter. *International Journal on Web Based Communities* 9 (2013), 122 – 139.

[5] Matthias Böhmer, Brent Hecht, Johannes Schöning, Antonio Krüger, and Gernot Bauer. 2011. Falling Asleep with Angry Birds, Facebook and Kindle: A Large Scale Study on Mobile Application Usage. In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services (MobileHCI '11)*. ACM, New York, NY, USA, 47–56.

[6] Matthias Böhmer and Antonio Krüger. 2013. A Study on Icon Arrangement by Smartphone Users. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, New York, NY, USA, 2137–2146.

[7] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. 2015. *Time series analysis: forecasting and control*. John Wiley & Sons.

[8] Andrew P Bradley. 1997. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern recognition* 30, 7 (1997), 1145–1159.

[9] Peter J Brockwell and Richard A Davis. 2013. *Time series: theory and methods*. Springer Science & Business Media.

[10] Mario Cataldi, Luigi Di Caro, and Claudio Schifanella. 2010. Emerging Topic Detection on Twitter Based on Temporal and Social Terms Evaluation. In *Proceedings of the Tenth International Workshop on Multimedia Data Mining*. 1–10.

[11] Rishi Chandy and Haijie Gu. 2012. Identifying Spam in the iOS App Store. In *Proceedings of the 2nd Joint WICOW/AIRWeb Workshop on Web Quality (WebQuality '12)*. ACM, New York, NY, USA, 56–59.

[12] Ning Chen, Jialiu Lin, Steven C. H. Hoi, Xiaokui Xiao, and Boshen Zhang. 2014. AR-miner: Mining Informative Reviews for Developers from Mobile App Marketplace. In *Proceedings of the 36th International Conference on Software Engineering (ICSE)*. 767–778.

[13] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. 2010. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the fourth ACM conference on Recommender systems*. 39–46.

[14] Mark de Reuver, Harry Bouwman, Nico Heerschap, and Hannu Verkasalo. 2012. Smartphone Measurement: do People Use Mobile Applications as they Say they do?. In *Proc. International Conference on Mobile Business (ICMB)*. 2–13.
[15] Bin Fu, Jialiu Lin, Lei Li, Christos Faloutsos, Jason I. Hong, and Norman M. Sadeh. 2013. Why people hate your app: making sense of user feedback in a mobile app store. In *Proceedings of The 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.
[16] Andrea Girardello and Florian Michahelles. 2010. AppAware: Which Mobile Applications Are Hot?. In *Proceedings of the 12th International Conference on Human Computer Interaction with Mobile Devices and Services (MobileHCI '10)*. ACM, New York, NY, USA, 431–434.
[17] Alessandra Gorla, Ilaria Tavecchia, Florian Gross, and Andreas Zeller. 2014. Checking App Behaviour Against App Descriptions. In *Proceedings of the 36th International Conference on Software Engineering (ICSE 2014)*. ACM, New York, NY, USA, 1025–1035.
[18] Emitza Guzman and Walid Maalej. 2014. How Do Users Like This Feature? A Fine Grained Sentiment Analysis of App Reviews. In *Proceedings of the IEEE International Requirements Engineering Conference (RE)*. 153–162.
[19] James Douglas Hamilton. 1994. *Time series analysis*. Vol. 2. Princeton university press Princeton.
[20] Mark Harman, Yue Jia, and Yuanyuan Zhang. 2012. App Store Mining and Analysis: MSR for App Stores. In *Proceedings of the 9th IEEE Working Conference on Mining Software Repositories (MSR '12)*. IEEE Press, 108–111.
[21] Xin Jin and Jiawei Han. 2010. *K-Means Clustering*. Springer US, Boston, MA, 563–564.
[22] Nitin Jindal and Bing Liu. 2008. Opinion Spam and Analysis. In *Proceedings of the 2008 International Conference on Web Search and Data Mining (WSDM '08)*. ACM, New York, NY, USA, 219–230.
[23] Alexandros Karatzoglou, Linas Baltrunas, Karen Church, and Matthias Böhmer. 2012. Climbing the App Wall: Enabling Mobile App Discovery Through Context-aware Recommendations. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management (CIKM '12)*. ACM, New York, NY, USA, 2527–2530.
[24] Hammad Khalid, Emad Shihab, Meiyappan Nagappan, and Ahmed E. Hassan. 2015. What Do Mobile App Users Complain About? *IEEE Software* 32 (2015), 70–77.
[25] Hee-Woong Kim, Hyun-Lyung Lee, and Su-Jin Choi. 2011. An exploratory study on the determinants of mobile application purchase. *The Journal of Society for e-Business Studies* 16, 4 (2011), 173–195.
[26] Yehuda Koren. 2010. Collaborative filtering with temporal dynamics. *Commun. ACM* 53, 4 (2010), 89–97.
[27] Neal Lathia, Stephen Hailes, Licia Capra, and Xavier Amatriain. 2010. Temporal Diversity in Recommender Systems. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '10)*. ACM, New York, NY, USA, 210–217.
[28] Soo Ling Lim, Peter J. Bentley, Nnatalie Kanakam, Fuyuki Ishikawa, and Shinichi Honiden. 2015. Investigating Country Differences in Mobile App User Behaviour and Challenges for Software Engineering. *IEEE Transactions on Software Engineering* 41, 1 (Jan 2015), 40–64.
[29] Jovian Lin, Kazunari Sugiyama, Min-Yen Kan, and Tat-Seng Chua. 2014. New and improved: modeling versions to improve app recommendation. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*. ACM, 647–656.
[30] Rui Liu, Jiannong Cao, Kehuan Zhang, Wenyu Gao, Lei Yang, Junbin Liang, and Ruiyun Yu. 2016. Understanding Mobile Users' Privacy Expectations: A Recommendation-based Method through Crowdsourcing. *IEEE Transactions on Services Computing* (December 2016).
[31] Michael Mathioudakis and Nick Koudas. 2010. Twittermonitor: trend detection over the twitter stream. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*. 1155–1158.
[32] Charles R. Nelson and Charles R. Plosser. 1982. Trends and random walks in macroeconmic time series: Some evidence and implications. *Journal of Monetary Economics* 10, 2 (1982), 139 – 162.
[33] Adam J. Oliner, Anand P. Iyer, Ion Stoica, Eemil Lagerspetz, and Sasu Tarkoma. 2013. Carat: Collaborative Energy Diagnosis for Mobile Devices. In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*. ACM, New York, NY, USA, Article 10, 14 pages.
[34] Dennis Pagano and Walid Maalej. 2013. User feedback in the appstore: An empirical study. In *Requirements Engineering Conference (RE), 2013 21st IEEE International*. IEEE, 125–134.
[35] Thanasis Petsas, Antonis Papadogiannakis, Michalis Polychronakis, Evangelos P. Markatos, and Thomas Karagiannis. 2013. Rise of the Planet of the Apps: A Systematic Study of the Mobile App Ecosystem. In *Proceedings of the 2013 Conference on Internet Measurement Conference (IMC '13)*. ACM, New York, NY, USA, 277–290.
[36] Pavel A. Pevzner. 2000. *Computational molecular biology – An algorithmic approach*. MIT Press.
[37] Alexandrin Popescul, Gary William Flake, Steve Lawrence, Lyle H Ungar, and C Lee Giles. 2000. Clustering and identifying temporal trends in document databases. In *Advances in Digital Libraries, 2000. Proceedings*. 173–182.
[38] Kira Radinsky, Krysta Svore, Susan Dumais, Jaime Teevan, Alex Bocharov, and Eric Horvitz. 2012. Modeling and predicting behavioral dynamics on the web. In *Proceedings of the 21st international conference on World Wide Web*. 599–608.

[39] Lenin Ravindranath, Jitendra Padhye, Sharad Agarwal, Ratul Mahajan, Ian Obermiller, and Shahin Shayandeh. 2012. AppInsight: Mobile App Performance Monitoring in the Wild.. In *OSDI*, Vol. 12. 107–120.

[40] Kent Shi and Kamal Ali. 2012. GetJar Mobile Application Recommendations with Very Sparse Datasets. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '12)*. ACM, New York, NY, USA, 204–212.

[41] Olga Streibel and Rehab Alnemr. 2011. Trend-based and Reputation-versed Personalized News Network. In *Proceedings of the 3rd International Workshop on Search and Mining User-generated Contents (SMUC '11)*. ACM, New York, NY, USA, 3–10.

[42] Hien Truong, Eemil Lagerspetz, Petteri Nurmi, Adam Oliner, Sasu Tarkoma, and N. Asokan. 2014. The company you keep: mobile malware infection rates and inexpensive risk indicators. In *Proceedings of the 23rd international conference on World wide web (WWW)*. ACM, 39 – 50.

[43] Sergei Vassilvitskii. 2007. *K-means: Algorithms, Analyses, Experiments*. Ph.D. Dissertation. Stanford, CA, USA. Advisor(s) Motwani, Rajeev.

[44] Fei Wang, Zhe Zhang, Hailong Sun, Richong Zhang, and Xudong Liu. 2013. A cooperation based metric for mobile applications recommendation. In *Web Intelligence (WI) and Intelligent Agent Technologies (IAT), 2013 IEEE/WIC/ACM International Joint Conferences on*, Vol. 3. IEEE, 13–16.

[45] Tingting Wang, Di Wu, Jiaming Zhang, Min Chen, and Yipeng Zhou. 2016. Measuring and Analyzing Third-Party Mobile Game App Stores in China. *IEEE Transactions on Network and Service Management* 13, 4 (Dec 2016), 793–805.

[46] Xiao Xia, Xiaodong Wang, Jian Li, and Xingming Zhou. 2014. Multi-objective mobile app recommendation: A system-level collaboration approach. *Computers & Electrical Engineering* 40, 1 (2014), 203–215.

[47] Ye Xu, Mu Lin, Hong Lu, Giuseppe Cardone, Nicholas Lane, Zhenyu Chen, Andrew Campbell, and Tanzeem Choudhury. 2013. Preference, context and communities: a multi-faceted approach to predicting smartphone app usage patterns. In *Proceedings of the 2013 International Symposium on Wearable Computers*. ACM, 69–76.

[48] Bo Yan and Guanling Chen. 2011. AppJoy: Personalized Mobile Application Discovery. In *Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services (MobiSys '11)*. ACM, New York, NY, USA, 113–126.

[49] Peifeng Yin, Ping Luo, Wang-Chien Lee, and Min Wang. 2013. App recommendation: a contest between satisfaction and temptation. In *Proceedings of the sixth ACM international conference on Web search and data mining*. ACM, 395–404.

[50] Nan Zhong and Florian Michahelles. 2013. Google play is not a long tail market: an empirical analysis of app adoption on the Google play app market. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing*. 499–504.

[51] Hengshu Zhu, Chuanren Liu, Yong Ge, Hui Xiong, and Enhong Chen. 2015. Popularity modeling for mobile apps: A sequential approach. *IEEE transactions on cybernetics* 45, 7 (2015), 1303–1314.

[52] Upasna Bhandari, Kazunari Sugiyama, Anindya Datta, and Rajni Jindal. 2013. Serendipitous recommendation for mobile apps using item-item similarity graph. *Asia Information Retrieval Symposium*. 440–451.

[53] Zhung-Xun Liao, Wen-Chih Peng, and S Yu Philip. 2013. Mining usage traces of mobile apps for dynamic preference prediction. *Asia-Pacific Conference on Knowledge Discovery and Data Mining*. 339-353.

Received February 2007; revised March 2009; accepted June 2009